|  |
| --- |
| Softura Solutions |
| Entregable PHP Laravel |

|  |  |
| --- | --- |
| **Proyecto:** | **Entregable PHP Laravel** |
| **Id. Documento:** | **1** |
| **Fecha:** | **11/09/22** |

# Instalando Herramientas y Configuración de Base de Datos

Para la realización de este proyecto se consideraron los siguientes requerimientos:

* Wampserver
* Composer
* Laravel 5.5
* PHP 7.2
* MySQL 5
* Apache 2.4.41

La instalación de Laravel se realizó a través de composer en línea de comandos del sistema, utilizando el siguiente comando:

* Utilizando composer
  + composer create-project --prefer-dist laravel/laravel nuevo-proyecto "5.5.\*“

Una vez realizado nuestro proyecto de laravel, se localiza la carpeta para posteriormente realizar nuestro Virtual Host de la siguiente manera:

Captura de pantalla de computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Una vez listo nuestro proyecto y Virtual Host procedemos a configurar nuestra base de datos que se ocupara durante el proyecto:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteEn este caso para el proyecto la base de datos creada fue :

curso\_laravel\_0822

Posterior a este paso procedemos a configurar a base de datos dentro del proyecto de laravel.

Captura de pantalla de computadora

Descripción generada automáticamente

En la carpeta config/database.php , vemos que podemos conectarnos por medio de varios gestores de base de datos pero configuraremos el que emplearemos en este caso MySQL , entonces colocamos los datos de importancia como el HOST, el PUERTO, el nombre de la base de datos , el usuario y la contraseña para acceder a esta.

Captura de pantalla de computadora

Descripción generada automáticamente

En el archivo .env colocamos la misma configuración para que exista comunicación con nuestra base de datos.

# 2. Creando LOGIN y Primeros paso del CRUD

Para crear la parte del Login nos dirigimos a terminal de nuestro IDE para poner el siguiente comando en línea de comandos:

Captura de pantalla de un celular

Descripción generada automáticamente

Luego de crear el login , realizamos la migraciones para crear nuestras tablas (´users´ , ‘password’) en nuestra base de datos:

Captura de pantalla de computadora

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Una vez creado nuestro login procedemos a crear el CRUD para empleados:

Captura de pantalla de un celular

Descripción generada automáticamente

Generamos nuestro modelo para identificar los campos que serán cubiertos en la base de datos:



En el modelo que se encontrara en la capteta app/Empleado.php realizamos el llenado del array $fillable de la siguiente manera:

Captura de pantalla de computadora

Descripción generada automáticamente

Y el controlador en el cual estarán definidas las funciones que ejecutara nuestro CRUD:



Captura de pantalla de computadora

Descripción generada automáticamente

# 3. Diseño del FRONT , RUTAS y funciones de CRUD

Texto

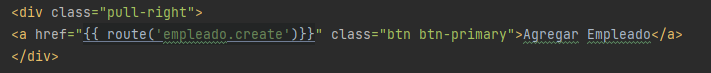
Descripción generada automáticamente

En la carpeta resources/view podremos manejar las vistas que se podrán ejecutar en el proceso del proyecto, creamos una carptella llamada Empleado donde agregaremos cada una de las vistas a emplear y comenzamos con el index.blade.php.

Texto

Descripción generada automáticamente

En el index lo que se busca es que los datos de los empleados sean visibles entonces creamos una tabla donde van a ser visualizados los datos y en donde en algunos botones agregamos el siguiente comando -> {{route}}:



Este código nos ayudara en el apartado de rutas:

Texto

Descripción generada automáticamente con confianza media

En el apartado de routes/web.php nosotros estableceremos las peticiones para controlar las vistas del CRUD y maneje los códigos de respuesta y métodos que requiere cada petición que se realice:



La ruta se realiza poniento el método , en este caso get, poniendo la estructura que tendrá la url , el controlador que en este caso es el EmpleadoController y el nombre de la ruta que es ‘empleado.create’: Captura de pantalla de computadora

Descripción generada automáticamente

Ahora continuamos con la pantalla de crear usuario, en el cual el procedimiento es parecido:

Texto

Descripción generada automáticamente

Es parecido , solo que aquí creamos un form que dirige a una nueva ruta que es .store , ahí se mandaran los datos para sus respectivas validaciones :



En la siguiente parte, dentro del controlador realizamos el traspaso de la información que fue obtenida del formulario a la base de datos realizando validaciones , como por ejemplo , los campos que son requeridos para que el formulario esté completo, los tipos de datos que son requeridos , si algún campo se requiere que sea dato único para que se realice el guardado de los datos.

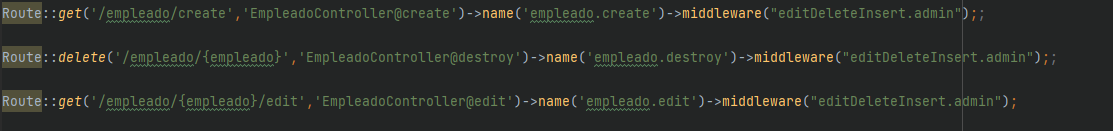
Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Las demás funciones son similares en Editar y Eliminar se trae el id del empleado para mostrar sus datos al momento de editar o para eliminar los datos del empleado:



Texto

Descripción generada automáticamente

En el caso de la función edit se realizario otra vista , agregando una ruta para la vista y para la ejecución de la modificación:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

# 4. Middleware y Lenguaje

Para la creación del Middleware se tiene la idea de que el CRUD no se active a menos que un usuario realizado en la creación del auth previamente visto acceda a su cuenta y/o este sea el administrador del CRUD:

Captura de pantalla de computadora

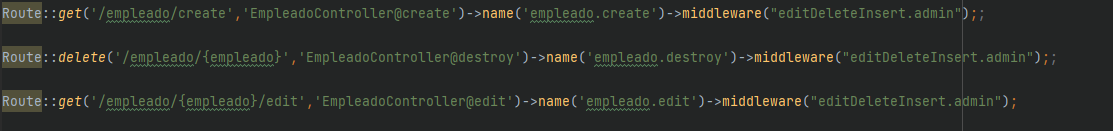
Descripción generada automáticamente

Para agregarlo a la aplicación necesitamos configurar la nueva clase a nuestro kernel:

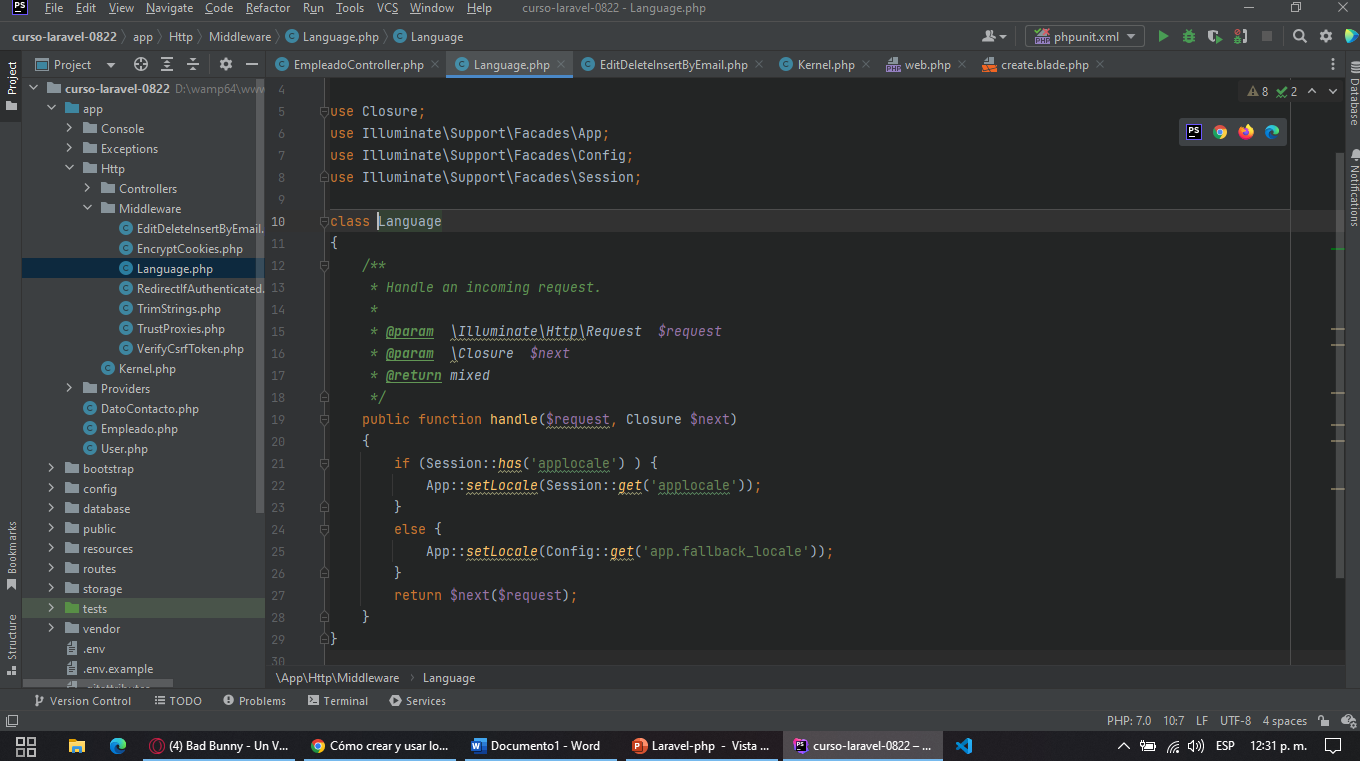
Una captura de pantalla de una computadora

Descripción generada automáticamente

Agregamos el middleware en las rutas que nos es de interes:



Para crear el cambio de lenguaje , realizamos igual la creación de nuestro Middleware pero en este caso nombre Languaje.php:



Aquí lo que se busca es que obtenga los datos que se encuentran en la configuración de la app:

Interfaz de usuario gráfica

Descripción generada automáticamente

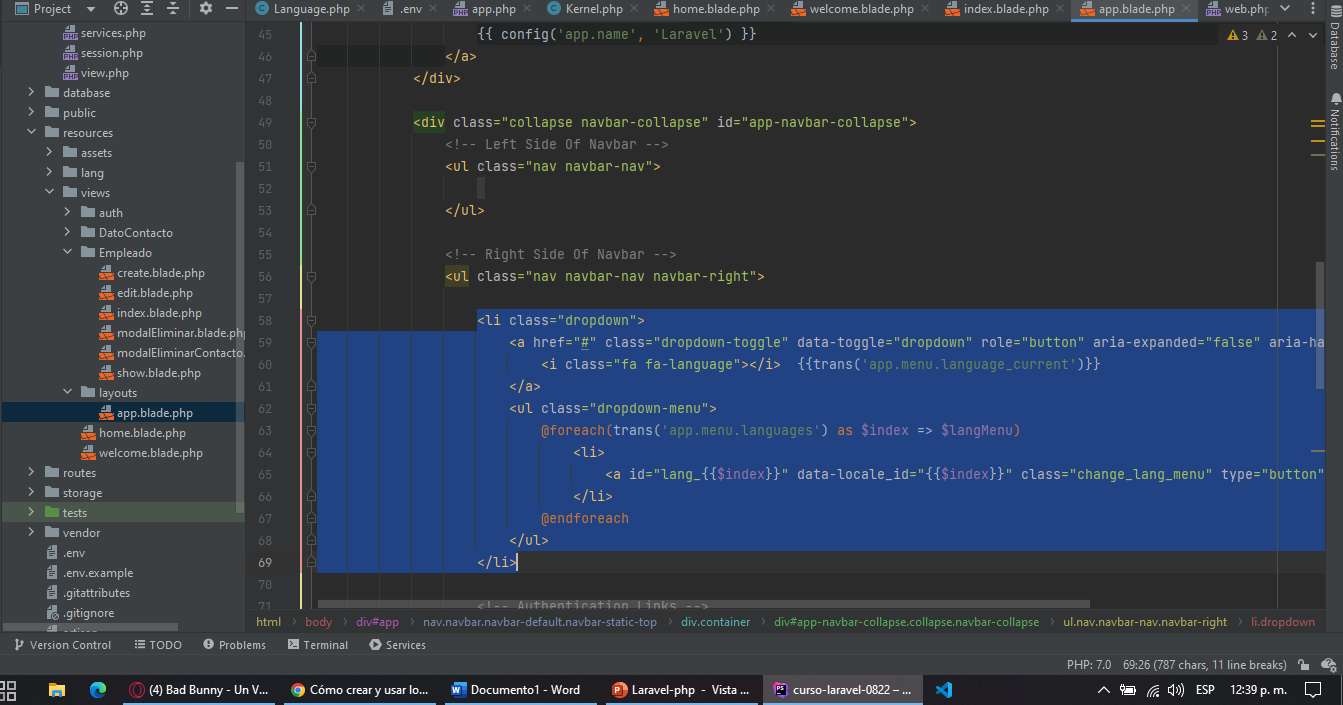
Manejamos el lenguaje que nos interesa en este caso el ingles (en) y el español(es) , de esta manera en el middleware la función es que si se selecciona que el locale sea español , el lenguaje de la App será en español o en caso contrario en ingles.

En el kernel se agrega la clase a ocupar , en este caso como no se manejaran rutas como tal el middleware se considera o implementa en el $middlewareGroups:

Captura de pantalla de computadora

Descripción generada automáticamente

Para complementar su funcionalidad, agregamos un componente en forma de dropdown en la plantilla de la cual se esta basando el CRUD:



En el cual especificamos que dentro de la barra de navegación se incluya un menú que pueda ejecutar el middleware para cambiar de idioma.

Creamos las carpetas en las cuales iran las traducciones a los idiomas que empleamos (español, ingles) en resources/lang.

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Una captura de pantalla de una computadora

Descripción generada automáticamente

Una captura de pantalla de una computadora

Descripción generada automáticamente

# 5. Consumo de Web Service

Para el consumo de web service se tomo en cuenta implementar en el CRUD en la sección de Agregar y Editar 2 nuevos campos: “Salario” y “tipo\_moneda” en el cual en el segundo campo se implementaría el Web Service:

Captura de pantalla de computadora

Descripción generada automáticamente

Se tuvo que realizar una nueva migración primero en línea de comando agregando el nombre de la tabla sobre la cual se trabajaria:



Se realizo la migración con :php artisan migrate agregando los campos con su respectivo up() y down()

Para obtener los datos del web service , en el controlador creamos nuestra función obtenerCurrencyWS

Texto

Descripción generada automáticamente

En nuestra función importamos la librería de Guzzle:

En composer :

* + composer require guzzlehttp/guzzle

En nuestro EmpleadoController

* + use GuzzleHttp\Client as HttpClient;

Se ejecuto en método GET para recibir los datos del Web Service y los decodificamos en XML debido a que se obtenían en XML los datos de las monedas.

Y tanto en las funciones de Agregar y Editar se implemento la función formando un arreglo con la función explode() para que formara todo el string en pequeños arreglos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para concluir en nuestro formulario colocamos un select en el cual mandamos a llamar la variable $listMonedas que contiene el arreglo de los tipo de monedas y lo implementamos de modo que se pueda escoger el que se requiere:

<select class="form-control" id="tipo\_moneda" name="tipo\_moneda">  
 <option value=""> >--Seleccione Tipo de Moneda--< </option>  
@foreach($listMonedas as $moneda)  
 <option value="{{$moneda}}">{{$moneda}}</option>  
@endforeach  
</select>

Ejecución:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente